Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Комп’ютерний практикум №8

Технології паралельних обчислень

Тема: Розробка алгоритмів для

розподілених систем клієнт-серверної архітектури

Виконав	Перевірила:
студент групи ІП-11:	Стеценко І.В.
Панченко С. В.	

Київ 2024

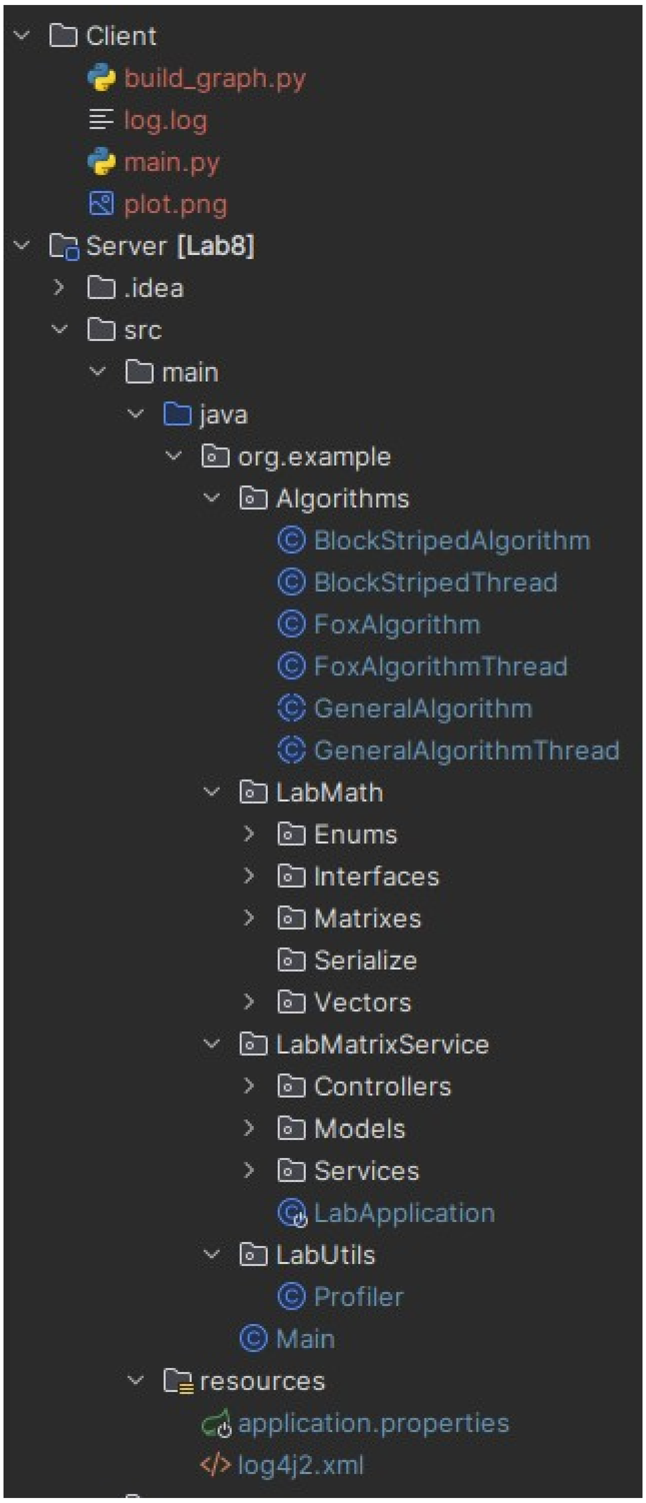
ЗМІСТ

1 ЗАВДАННЯ

1. Розробити веб-застосування клієнт-серверної архітектури, що реалізує алгоритм множення матриць або інший, який був Вами реалізований в рамках курсу «Технології паралельних обчислень», на стороні сервера з використанням паралельних обчислень. Розгляньте два варіанти реалізації 1) дані для обчислень знаходяться на сервері та 2) дані для обчислень знаходяться на клієнтській частині застосування. 60 балів.
2. Дослідити швидкість виконання запиту користувача при різних обсягах даних. 30 балів.
3. Порівняти реалізацію алгоритму в клієнт-серверній системі та в розподіленій системі з рівноправними процесорами. 10 балів.

2 ВИКОНАННЯ

2.1 Структура проєкту



```
Client
    build_graph.py
    log.log
    main.py
    plot.png
Server [Lab8]
  .idea
  src
    main
      java
        org.example
          Algorithms
            BlockStripedAlgorithm
            BlockStripedThread
            FoxAlgorithm
            FoxAlgorithmThread
            GeneralAlgorithm
            GeneralAlgorithmThread
          LabMath
            Enums
            Interfaces
            Matrixes
            Serialize
            Vectors
          LabMatrixService
            Controllers
            Models
            Services
            LabApplication
          LabUtils
            Profiler
          Main
      resources
        application.properties
        log4j2.xml
```

Рисунок 2.1.1 - Структура проєкту

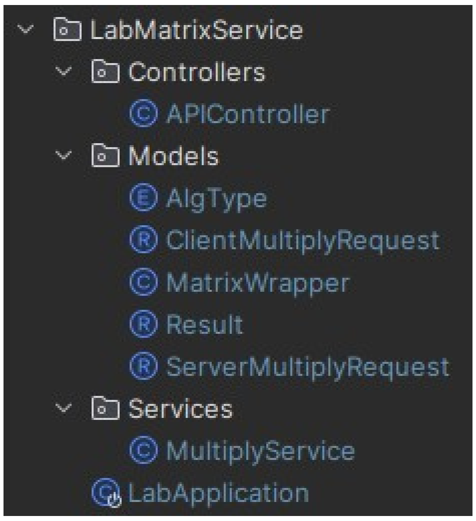
Загально проєкт складається з двох частин: Client та Server.

2.2 Структура серверу

Загалом сервер складається з математичної бібліотеки та алгоритмів, які були описані в минулих комп’ютерних практикумах, тому сконцентруємося на головному.

LabUtils — модуль, що включає в себе додаткові допоміжні класи, як-от Profiler, що вимірює час виконання функції.

Поглянемо на LabMatrixService.



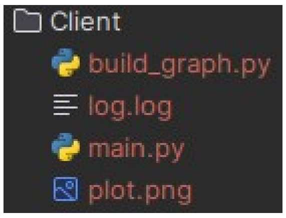
```
LabMatrixService
  Controllers
    APIController
  Models
    AlgType
    ClientMultiplyRequest
    MatrixWrapper
    Result
    ServerMultiplyRequest
  Services
    MultiplyService
  LabApplication
```

Controllers — API-контролери, що реагують на запити та викликають сервіси для користувача.

Models — модуль, що включає в себе перелік типів аргументів, класи запитів, обгортки над класом матриці тощо.

Services — модуль, що включає сервіс множення матриць.

2.3 Структура клієнту



```
Client
  build_graph.py
  log.log
  main.py
  plot.png
```

Рисунок 2.3.1 - Структура клієнту

Складається з скрипту main.py, що використовує модуль requests для надсилання запитів на сервер та логування результатів, а build.py — будує графік.

2.4 Результати

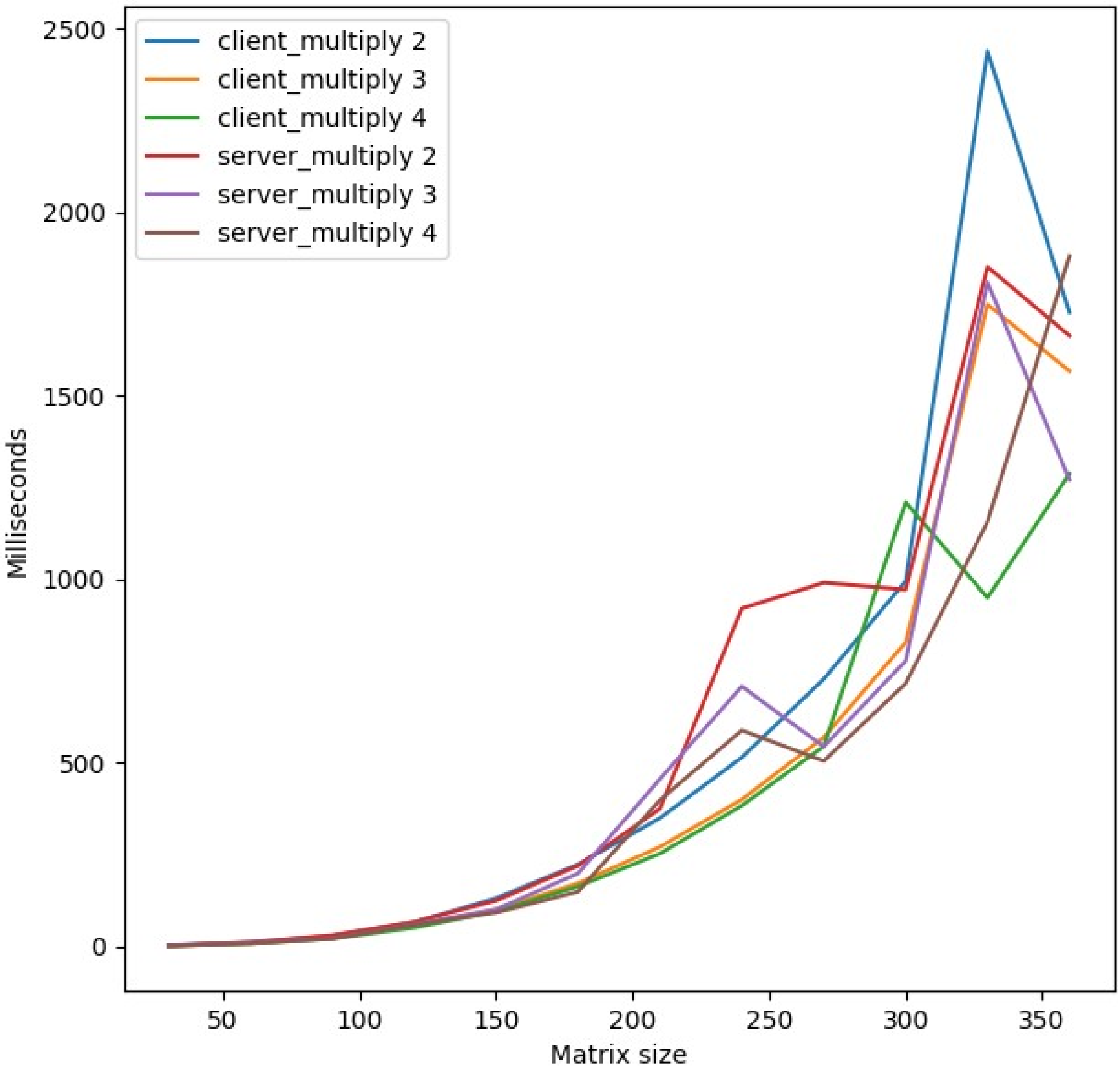


Рисунок 2.4.1 - Графіки результатів

Як бачимо, що коли дані для обчислень знаходяться на клієнті (client_multiply), то запити виконуються довше, оскільки потрібні також надіслати матриці у запиті, що очевидно довше.

3 ВИСНОВОК

Під час лабораторної роботи опрацювали розробка алгоритмів для розподілених систем клієнт-серверної архітектури.

Побудували графік та показали, що при надсиланні даних від користувача, запити виконуються довше.

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду

(Найменування програми (документа))

Жорсткий диск

(Вид носія даних)

(Обсяг програми (документа), арк.)

Студента групи ІП-113 курсу

Панченка С. В

```
// ./Lab8/Server/src/main/java/org/example/Main.java

package org.example;

// Press Shift twice to open the Search Everywhere dialog and type `show
whitespaces`,
// then press Enter. You can now see whitespace characters in your code.
public class Main {
        public static void main(String[] args) {
                // Press Alt+Enter with your caret at the highlighted text to see how
                // IntelliJ IDEA suggests fixing it.
                System.out.printf("Hello and welcome!");

                // Press Ctrl+F5 or click the green arrow button in the gutter to run the
code.
                for(int i = 1; i <= 5; i++) {

                        // Press Alt+F5 to start debugging your code. We have set one
breakpoint
                        // for you, but you can always add more by pressing F9.
                        System.out.println("i = " + i);
                }
        }
}

// ./Lab8/Server/src/main/java/org/example/LabUtils/Profiler.java

package org.example.LabUtils;

import javafx.util.Pair;
import java.util.concurrent.Callable;
```

```
public class Profiler {
	public <T> Pair<T, Long> performBenchmark(Callable<T> func) throws Exception {
		var startTime = System.nanoTime();
		var result = func.call();
		var duration = System.nanoTime() - startTime;
		return new Pair<>(result, duration);
	}
}


// ./Lab8/Server/src/main/java/org/example/LabMatrixService/LabApplication.java

package org.example.LabMatrixService;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class LabApplication {
	public static void main(String[] args) {
		SpringApplication.run(LabApplication.class, args);
	}
}


// ./Lab8/Server/src/main/java/org/example/LabMatrixService/Controllers/APIController.java

package org.example.LabMatrixService.Controllers;

import org.example.LabMath.Matrixes.Matrix2DFactory;
import org.example.LabMatrixService.Models.*;
import org.example.LabMatrixService.Services.MultiplyService;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api")
public class APIController {
        private static final int MIN_VAL = 0;
        private static final int MAX_VAL = 1;
        private static final int DEFAULT_SEED = 0;

        @GetMapping("/clientMultiply")
        public ResponseEntity<Result> clientMultiply(@RequestBody
ClientMultiplyRequest request) throws Exception {
                var result = MultiplyService.solve(
                        request.algType(), request.threadsNum(),
                        request.first().getMat(), request.second().getMat());
                return ResponseEntity.ok(result);
        }

        @GetMapping("/serverMultiply")
        public ResponseEntity<Result> serverMultiply(
                @RequestBody ServerMultiplyRequest request) throws Exception {

                var factory = new Matrix2DFactory();
                var first = factory.getRandom(request.rows(), request.cols(),
                        MIN_VAL, MAX_VAL, DEFAULT_SEED);
                var second = factory.getRandom(request.rows(), request.cols(),
                        MIN_VAL, MAX_VAL, DEFAULT_SEED);
                var result = MultiplyService.solve(
                        request.algType(), request.threadsNum(), first, second);
                return ResponseEntity.ok(result);
        }
}
```

// ./Lab8/Server/src/main/java/org/example/LabMatrixService/Services/MultiplyService.java

```
package org.example.LabMatrixService.Services;

import org.example.Algorithms.BlockStripedAlgorithm;
import org.example.Algorithms.FoxAlgorithm;
import org.example.LabMatrixService.Models.AlgType;
import org.example.LabMatrixService.Models.Result;
import org.example.LabMath.Matrixes.Matrix2D;
import org.example.LabUtils.Profiler;

public class MultiplyService {
    public MultiplyService() {}

    public static Result solve(AlgType algType, int threadsNum, Matrix2D first, Matrix2D second) throws Exception {
        var data = new Profiler().performBenchmark(() -> switch(algType) {
                case BLOCK_STRIPED -> new BlockStripedAlgorithm(threadsNum, first, second).solve();
                case FOX -> new FoxAlgorithm(threadsNum, first, second).solve();
                case NATIVE -> first.getMul(second);
            }
        );
        return new Result(algType, threadsNum, data.getValue());
    }
}

// ./Lab8/Server/src/main/java/org/example/LabMatrixService/Models/Result.java

package org.example.LabMatrixService.Models;

public record Result(AlgType algType, int threadsNum, long nanoseconds) {
}


//
./Lab8/Server/src/main/java/org/example/LabMatrixService/Models/ClientMultiplyR
equest.java

package org.example.LabMatrixService.Models;

public record ClientMultiplyRequest(AlgType algType, MatrixWrapper first,
MatrixWrapper second, int threadsNum) {
}


// ./Lab8/Server/src/main/java/org/example/LabMatrixService/Models/AlgType.java

package org.example.LabMatrixService.Models;

public enum AlgType {
    BLOCK_STRIPED,
    FOX,
    NATIVE
```

```
}


//
./Lab8/Server/src/main/java/org/example/LabMatrixService/Models/ServerMultiplyR
equest.java

package org.example.LabMatrixService.Models;

public record ServerMultiplyRequest(AlgType algType, int rows, int cols, int seed,
int threadsNum) {
}


//
./Lab8/Server/src/main/java/org/example/LabMatrixService/Models/MatrixWrapper.j
ava

package org.example.LabMatrixService.Models;

import org.example.LabMath.Matrixes.GeneralMatrix;
import org.example.LabMath.Matrixes.Matrix2D;

public class MatrixWrapper {
    private final Matrix2D mat;
    public MatrixWrapper(int rows, int cols, double[] values) {
        mat = new Matrix2D(rows, cols);
        for(var i = 0; i < rows; ++i) {
            for(var j = 0; j < cols; ++j) {
                mat.setAt(values[i * cols + j], i, j);
            }
        }
        }
        public Matrix2D getMat() {
                return mat;
        }
}


// ./Lab8/Server/src/main/java/org/example/LabMath/Enums/Coords.java

package org.example.LabMath.Enums;

public enum Coords {
   Y,
   X
}


// ./Lab8/Server/src/main/java/org/example/LabMath/Vectors/GeneralVector.java

package org.example.LabMath.Vectors;

import org.example.LabMath.Interfaces.MathVector;

import java.util.Arrays;

public class GeneralVector implements MathVector<GeneralVector> {
      private static final String ERROR_LENGTHS_NOT_EQUAL = "Lengths of
points arguments are not equal";
   private double[] arguments;

   public GeneralVector(int length) {
      setLength(length);
```

```
}

public GeneralVector(GeneralVector other) {
   setLength(other.getLength());
   set(other);
}

public int getLength() {
   if(arguments == null) {
      return 0;
   }
   return arguments.length;
}

public void setLength(int length) {
   var currentLength = getLength();

   if(currentLength==length) return;

   var minLength = Math.min(currentLength, length);
   var args = new double[length];
   for(var i = 0; i < minLength; ++i) {
      args[i] = getAt(i);
   }

   arguments = args;
}

@Override
public void set(GeneralVector other) {
   checkSizesEqual(other);
   for(var i = 0; i < getLength(); ++i) {
        setAt(i, other.getAt(i));
    }
}

@Override
public GeneralVector clone() {
    return new GeneralVector(this);
}

@Override
public String toString() {
    return Arrays.toString(arguments);
}

private void checkSizesEqual(GeneralVector other) {
    assert getLength() == other.getLength() : ERROR_LENGTHS_NOT_EQUAL;
}

@Override
public void add(GeneralVector other) {
    checkSizesEqual(other);
    for(var i = 0; i < getLength(); ++i) {
        setAt(i, getAt(i) + other.getAt(i));
    }
}

@Override
public void add(double value) {
    for(var i = 0; i < getLength(); ++i) {
        setAt(i, getAt(i) + value);
    }
}
```

```
@Override
public void sub(GeneralVector other) {
   checkSizesEqual(other);
   for(var i = 0; i < getLength(); ++i) {
      setAt(i, getAt(i) - other.getAt(i));
   }
}

@Override
public void sub(double value) {
   for(var i = 0; i < getLength(); ++i) {
      setAt(i, getAt(i) - value);
   }
}

@Override
public void mul(GeneralVector other) {
   checkSizesEqual(other);
   for(var i = 0; i < getLength(); ++i) {
      setAt(i, getAt(i) * other.getAt(i));
   }
}

@Override
public void mul(double value) {
   for(var i = 0; i < getLength(); ++i) {
      setAt(i, getAt(i) * value);
   }
}

@Override
```

```
public void div(GeneralVector other) {
   checkSizesEqual(other);
   for(var i = 0; i < getLength(); ++i) {
      setAt(i, getAt(i) / other.getAt(i));
   }
}

@Override
public void div(double value) {
   for(var i = 0; i < getLength(); ++i) {
      setAt(i, getAt(i) / value);
   }
}

@Override
public double getSize() {
   return Math.sqrt(getSizeSquared());
}

@Override
public double getSizeSquared() {
   double s = 0;
   for(var i = 0; i < getLength(); ++i) {
      s += Math.pow(getAt(i), 2);
   }
   return s;
}

@Override
public double getDotProduct(GeneralVector other) {
   checkSizesEqual(other);
   var prod = 0;
```

```
    for(var i = 0; i < getLength(); ++i) {
        prod += getAt(i) * other.getAt(i);
    }
    return prod;
}

@Override
public double getDistance(GeneralVector other) {
    checkSizesEqual(other);
    var dist = 0.0;
    for(var i = 0; i < getLength(); ++i) {
        dist += Math.pow(getAt(i) - other.getAt(i), 2);
    }
    dist = Math.sqrt(dist);
    return dist;
}

@Override
public GeneralVector getForwardVector() {
    var forwardVec = clone();
    var size = getSize();
    for(var i = 0; i < getLength(); ++i) {
        forwardVec.setAt(i, getAt(i) / size);
    }
    return forwardVec;
}

@Override
public double getAt(int index) {
    return arguments[index];
}
```

```
    @Override
    public void setAt(int index, double value) {
        arguments[index] = value;
    }

    @Override
    public GeneralVector getOpposite() {
        var v = clone();
        v.toOpposite();
        return v;
    }

    @Override
    public void toOpposite() {
        for(var i = 0; i < getLength(); ++i) {
            setAt(i, -getAt(i));
        }
    }

}


// ./Lab8/Server/src/main/java/org/example/LabMath/Vectors/Vector2D.java

package org.example.LabMath.Vectors;

import org.example.LabMath.Enums.*;
import org.example.LabMath.Interfaces.MathVector;

public class Vector2D implements MathVector<Vector2D> {
    private final GeneralVector vec = new GeneralVector(2);
```

```
public Vector2D() {}

public Vector2D(double y, double x) {
   set(y, x);
}

public Vector2D(Vector2D other) {
   set(other);
}

public double getX() {
   return getAt(Coords.X.ordinal());
}

public double getY() {
   return getAt(Coords.Y.ordinal());
}

public void set(Vector2D other) {
   vec.set(other.vec);
}

public void set(double y, double x) {
   setX(x);
   setY(y);
}

public void setX(double value) {
   setAt(Coords.X.ordinal(), value);
}

public void setY(double value) {
    setAt(Coords.Y.ordinal(), value);
}

@Override
public Vector2D clone() {
    return new Vector2D(getY(), getX());
}

@Override
public String toString() {
    return vec.toString();
}

@Override
public void add(Vector2D other) {
    vec.add(other.vec);
}

@Override
public void add(double value) {
    vec.add(value);
}

@Override
public void sub(double value) {
    vec.sub(value);
}

@Override
public void sub(Vector2D other) {
    vec.sub(other.vec);
}
```

```
@Override
public void mul(Vector2D other) {
   vec.mul(other.vec);
}

@Override
public void mul(double value) {
   vec.mul(value);
}

@Override
public void div(Vector2D other) {
   vec.div(other.vec);
}

@Override
public void div(double value) {
   vec.div(value);
}

@Override
public double getSize() {
   return vec.getSize();
}

@Override
public double getSizeSquared() {
   return vec.getSizeSquared();
}

@Override
```

```
public double getDotProduct(Vector2D other) {
    return vec.getDotProduct(other.vec);
}

@Override
public double getDistance(Vector2D other) {
    return vec.getDistance(other.vec);
}

@Override
public Vector2D getForwardVector() {
    var forwardVec = clone();
    forwardVec.vec.set(forwardVec.vec.getForwardVector());
    return forwardVec;
}

@Override
public double getAt(int index) {
    return vec.getAt(index);
}

@Override
public void setAt(int index, double value) {
    vec.setAt(index, value);
}

@Override
public Vector2D getOpposite() {
    var v = clone();
    v.toOpposite();
    return v;
}
```

```
    @Override
    public void toOpposite() {
        vec.toOpposite();
    }
}


// ./Lab8/Server/src/main/java/org/example/LabMath/Interfaces/MathVector.java

package org.example.LabMath.Interfaces;

import org.example.LabMath.Interfaces.General.*;

public interface MathVector<T> extends Cloneable, Divisible<T>, Multipliable<T>,
Addable<T>, Subtractable<T>,
        DoubleDivisible, DoubleMultipliable, DoubleAddable, DoubleSubtractable {
    double getSize();
    double getSizeSquared();
    double getDotProduct(T other);
    double getDistance(T other);
    T getForwardVector();
    double getAt(int index);
    void setAt(int index, double value);
    T getOpposite();
    void toOpposite();
    void set(T other);
}


// ./Lab8/Server/src/main/java/org/example/LabMath/Interfaces/MathMatrix.java
```

```
package org.example.LabMath.Interfaces;

import org.example.LabMath.Interfaces.General.*;

public interface MathMatrix<T> extends Cloneable, Addable<T>, Subtractable<T>, 
Divisible<T>,
        GetMultipliable<T>, Settable<T>, DoubleSubtractable, DoubleMultipliable, 
DoubleAddable, DoubleDivisible {
    int[] getDimensions();
    double getAt(int... indexes);
    void setAt(double value, int... indexes);
    int calcIndex(int... indexes);
}


//                ./Lab8/Server/src/main/java/org/example/LabMath/Interfaces/General/
DoubleDivisible.java

package org.example.LabMath.Interfaces.General;

public interface DoubleDivisible {
    void div(double other);
}


//                ./Lab8/Server/src/main/java/org/example/LabMath/Interfaces/General/
Divisible.java

package org.example.LabMath.Interfaces.General;

import jdk.jshell.spi.ExecutionControl;
```

```
public interface Divisible<T> {
   void div(T other) throws ExecutionControl.NotImplementedException;
}


// ./Lab8/Server/src/main/java/org/example/LabMath/Interfaces/General/Addable.java

package org.example.LabMath.Interfaces.General;

public interface Addable<T> {
   void add(T other);
}


// ./Lab8/Server/src/main/java/org/example/LabMath/Interfaces/General/Subtractable.java

package org.example.LabMath.Interfaces.General;

public interface Subtractable<T> {
   void sub(T other);
}


// ./Lab8/Server/src/main/java/org/example/LabMath/Interfaces/General/DoubleAddable.java

package org.example.LabMath.Interfaces.General;

public interface DoubleAddable {
   void add(double other);
}
```

```
//                    ./Lab8/Server/src/main/java/org/example/LabMath/Interfaces/General/
DoubleSubtractable.java

package org.example.LabMath.Interfaces.General;

public interface DoubleSubtractable {
   void sub(double other);
}


// ./Lab8/Server/src/main/java/org/example/LabMath/Interfaces/General/Settable.java

package org.example.LabMath.Interfaces.General;

public interface Settable<T> {
   void set(T other);
}


//                    ./Lab8/Server/src/main/java/org/example/LabMath/Interfaces/General/
DoubleMultipliable.java

package org.example.LabMath.Interfaces.General;

public interface DoubleMultipliable {
   void mul(double other);
}


//                    ./Lab8/Server/src/main/java/org/example/LabMath/Interfaces/General/
```

```
Multipliable.java

package org.example.LabMath.Interfaces.General;

public interface Multipliable<T> {
    void mul(T other);
}


//                  ./Lab8/Server/src/main/java/org/example/LabMath/Interfaces/General/
GetMultipliable.java

package org.example.LabMath.Interfaces.General;

import jdk.jshell.spi.ExecutionControl;

public interface GetMultipliable<T> {
   T getMul(T other) throws ExecutionControl.NotImplementedException;
}


// ./Lab8/Server/src/main/java/org/example/LabMath/Matrixes/Matrix2DFactory.java

package org.example.LabMath.Matrixes;

import java.util.Random;

public class Matrix2DFactory {

    public Matrix2DFactory() {}

    public static void main(String[] args) {
        var factory = new Matrix2DFactory();
        var minVal = 0;
        var maxVal = 10;
        var rows = 5;
        var cols = 6;
        var one = factory.getRandom(rows, cols, minVal, maxVal, 0);
        var two = factory.getRandom(cols, rows, minVal, maxVal, 0);
        var result = one.getMul(two);
        System.out.println(result);
    }

    public Matrix2D getRandom(int rows, int cols, int minVal, int maxVal, int seed) {
        var random = new Random(seed);
        var res = new Matrix2D(rows, cols);
        for(var i = 0; i < rows; ++i) {
            for(var j = 0; j < cols; ++j) {
                res.setAt(random.nextDouble() * (maxVal - minVal) + minVal, i, j);
            }
        }
        return res;
    }
}


// ./Lab8/Server/src/main/java/org/example/LabMath/Matrixes/GeneralMatrix.java

package org.example.LabMath.Matrixes;

import jdk.jshell.spi.ExecutionControl;
import org.example.LabMath.Interfaces.MathMatrix;
import org.example.LabMath.Vectors.GeneralVector;
import java.util.Arrays;

public final class GeneralMatrix implements MathMatrix<GeneralMatrix> {
   private static final String ERROR_INDEXES = "Indexes are less than 0";
      private static final String ERROR_DIMENSIONS = "Matrix dimensions not 
equal";
    private static final String ERROR_DIMENSION_INDEXES = "Indexes length is 
not equal to amount of dimension";
   private final int[] dimensions;
   private final int total;
   private final GeneralVector mat;

   public GeneralMatrix(int... dimensions) {
      this.dimensions = dimensions.clone();
      var t = 1;
      for(var d : dimensions) t *= d;
      this.total = t;
      this.mat = new GeneralVector(this.total);
   }

   private String doDraw(int[] indexes, int dimension) {
      var res = new StringBuilder();
      res.append("{");
      for(var i = 0; i < this.dimensions[dimension]; ++i) {
         indexes[dimension] = i;
         if(dimension == this.dimensions.length - 1) {
            res.append(this.mat.getAt(calcIndex(indexes)));
         } else {
            res.append(doDraw(indexes, dimension + 1));
         }
         res.append(this.dimensions[dimension] - 1 == i ? "" : ", ");
      }
    res.append("}");
    return res.toString();
}

@Override
public String toString() {
    var indexes = new int[this.dimensions.length];
    return doDraw(indexes, 0);
}

private void checkDimensions(int[] dimensions) {
    if(!Arrays.equals(this.dimensions, dimensions)) {
        throw new IllegalArgumentException(ERROR_DIMENSIONS);
    }
}

private void checkIndexes(int[] indexes) {
    if(!Arrays.stream(indexes).allMatch(e -> e >= 0)) {
        throw new IllegalArgumentException(ERROR_INDEXES);
    }
}

@Override
public void add(GeneralMatrix other) {
    checkDimensions(other.dimensions);
    for(var i = 0; i < total; ++i) {
        this.mat.setAt(i, this.mat.getAt(i) + other.mat.getAt(i));
    }
}

@Override
public void add(double value) {
        for(var i = 0; i < this.total; ++i) {
            this.mat.setAt(i, this.mat.getAt(i) + value);
        }
    }

    @Override
    public void div(double value) {
        for(var i = 0; i < this.total; ++i) {
            this.mat.setAt(i, this.mat.getAt(i) / value);
        }
    }

    @Override
    public void mul(double value) {
        for(var i = 0; i < this.total; ++i) {
            this.mat.setAt(i, this.mat.getAt(i) * value);
        }
    }

    @Override
    public void sub(double value) {
        for(var i = 0; i < this.total; ++i) {
            this.mat.setAt(i, this.mat.getAt(i) - value);
        }
    }

    @Override
    public GeneralMatrix getMul(GeneralMatrix other) throws
ExecutionControl.NotImplementedException {
        throw new ExecutionControl.NotImplementedException("");
    }
```

```
@Override
public void set(GeneralMatrix other) {
   checkDimensions(other.dimensions);
   for(var i = 0; i < this.total; ++i) {
      this.mat.setAt(i, this.mat.getAt(i));
   }
}

@Override
public void sub(GeneralMatrix other) {
   checkDimensions(other.dimensions);
   for(var i = 0; i < total; ++i) {
      this.mat.setAt(i, other.mat.getAt(i) - other.mat.getAt(i));
   }
}

@Override
public int[] getDimensions() {
   return dimensions.clone();
}

@Override
public double getAt(int... indexes) {
   checkIndexes(indexes);
   return this.mat.getAt(this.calcIndex(indexes));
}

@Override
public void setAt(double value, int... indexes) {
   checkIndexes(indexes);
   var index = this.calcIndex(indexes);
   this.mat.setAt(index, value);
```

```
    }

    @Override
                public    void    div(GeneralMatrix    other)    throws
ExecutionControl.NotImplementedException {
        throw new ExecutionControl.NotImplementedException("");
    }

    @Override
    public int calcIndex(int... indexes) {
        if(indexes.length != dimensions.length) {
            throw new IllegalArgumentException(ERROR_DIMENSION_INDEXES);
        }
        var index = 0;
        var mult = 1;
        for(var i : dimensions) mult *= i;
        for(var i = 0; i < indexes.length; ++i) {
            mult /= dimensions[i];
            index += indexes[i] * mult;
        }
        return index;
    }
}


// ./Lab8/Server/src/main/java/org/example/LabMath/Matrixes/Matrix2D.java

package org.example.LabMath.Matrixes;

import jdk.jshell.spi.ExecutionControl;
import org.example.LabMath.Interfaces.MathMatrix;
import org.springframework.web.multipart.MultipartFile;
```

```
public class Matrix2D implements MathMatrix<Matrix2D> {
    private static final String ERROR_MULTIPLICATION = "Rows and columns are
not equal";
   private static final String ERROR_INDEXES = "Indexes are less than 0";
   private final int rows;
   private final int cols;
   private final GeneralMatrix mat;

   public static void main(String[] args) {}

   @Override
   public String toString() {
      return mat.toString();
   }

   public Matrix2D(int rows, int cols) {
      mat = new GeneralMatrix(rows, cols);
      this.rows = rows;
      this.cols = cols;
   }

   public int getRows() {
      return this.rows;
   }

   public int getCols() {
      return this.cols;
   }

   @Override
   public void add(Matrix2D other) {
      this.mat.add(other.mat);
   }

   @Override
                           public        void        div(Matrix2D        other)        throws
ExecutionControl.NotImplementedException {
      throw new ExecutionControl.NotImplementedException("");
   }

   @Override
   public void add(double value) {
      this.mat.add(value);
   }

   @Override
   public void div(double value) {
      this.mat.div(value);
   }

   @Override
   public void mul(double value) {
      this.mat.mul(value);
   }

   @Override
   public void sub(double value) {
      this.mat.sub(value);
   }

   @Override
   public Matrix2D getMul(Matrix2D other) {
      var cols = getCols();
      assert cols == other.getRows() : ERROR_MULTIPLICATION;

      var result = new Matrix2D(rows, cols);

      for(var i = 0; i < getRows(); ++i) {
        for(var j = 0; j < other.getCols(); ++j) {
          var value = 0;
          for(var k = 0; k < cols; ++k) {
            value += this.mat.getAt(i, k) * other.mat.getAt(k, j);
          }
          result.setAt(value, i, j);
        }
      }

      return result;
   }

   @Override
   public void set(Matrix2D other) {
      this.mat.set(other.mat);
   }

   @Override
   public void sub(Matrix2D other) {
      this.mat.sub(other.mat);
   }

   @Override
   public int[] getDimensions() {
      return this.mat.getDimensions();
   }
```

```
    @Override
    public double getAt(int... indexes) {
        if(indexes.length != 2) {
            throw new IllegalArgumentException(ERROR_INDEXES);
        }
        return this.mat.getAt(indexes);
    }

    @Override
    public void setAt(double value, int... indexes) {
        if(indexes.length != 2) {
            throw new IllegalArgumentException(ERROR_INDEXES);
        }
        this.mat.setAt(value, indexes);
    }

    @Override
    public int calcIndex(int... indexes) {
        return this.mat.calcIndex(indexes);
    }

    public boolean isSquare() {
        return this.rows == this.cols;
    }
}


// ./Lab8/Server/src/main/java/org/example/Algorithms/BlockStripedAlgorithm.java

package org.example.Algorithms;
```

```
import org.example.LabMath.Matrixes.Matrix2D;
import org.example.LabMath.Matrixes.Matrix2DFactory;

public class BlockStripedAlgorithm extends GeneralAlgorithm {

    public BlockStripedAlgorithm() {}

     public BlockStripedAlgorithm(int threadsNum, Matrix2D first, Matrix2D second)
{
        super(threadsNum, first, second);
    }

    public static void main(String[] args) {
        var matrixFactory = new Matrix2DFactory();
        var rows = 10;
        var cols = 10;
        var minVal = 0;
        var maxVal = 10;
        var threadsNum = 5;
        var seed = 0;
        var first = matrixFactory.getRandom(rows, cols, minVal, maxVal, seed);
        var second = matrixFactory.getRandom(rows, cols, minVal, maxVal, seed);
        var algorithm = new BlockStripedAlgorithm(threadsNum, first, second);
        var result = algorithm.solve();
        System.out.println("First:\t" + first);
        System.out.println("Second:\t" + second);
        System.out.println("Result:\t" + result);
    }

    public Matrix2D solve() {
        var firstRows = first.getRows();
        var firstCols = first.getCols();
        var secondRows = second.getRows();
        var secondCols = second.getCols();

        if(firstCols != secondRows) {
            throw new IllegalArgumentException(ERROR_MULTIPLICATION);
        }

        var result = new Matrix2D(firstRows, secondCols);
        var isRowsLess = firstRows < threads.length;
        var totalThreads = isRowsLess ? firstRows : threads.length;
        var step = isRowsLess ? 1 : threads.length;

        for(var i = 0; i < totalThreads; ++i) {
            threads[i] = new BlockStripedThread(i, step, first, second, result);
        }

        for(var i = 0; i < totalThreads; ++i) {
            threads[i].start();
        }

        for(var i = 0; i < totalThreads; ++i) {
            try {
                threads[i].join();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        return result;
    }
}
```

```
// ./Lab8/Server/src/main/java/org/example/Algorithms/FoxAlgorithmThread.java

package org.example.Algorithms;

import org.example.LabMath.Matrixes.Matrix2D;

public class FoxAlgorithmThread extends GeneralAlgorithmThread {
    private final int start;
    private final int blockSize;

    public FoxAlgorithmThread(int index, int blockSize, Matrix2D first, Matrix2D second, Matrix2D result) {
        super(first, second, result);
        this.start = index * blockSize;
        this.blockSize = blockSize;
    }

    @Override
    public void run() {
        var curRow = start;
        var curCol = start;
        var rows = first.getRows();
        var steps = rows / blockSize;
        for(var i = 0; i < steps; ++i) {
            for(var j = 0; j < steps; ++j) {
                mulMatrices(curCol, curRow, j * blockSize);
            }
            curRow += blockSize;
            curRow %= rows;
            curCol += blockSize;
            curCol %= rows;
        }
    }

    public void mulMatrices(int firstCol, int secondRow, int secondCol) {
        for(var i = start; i < start + blockSize; ++i) {
            for(var j = secondCol; j < secondCol + blockSize; ++j) {
                var value = 0.0;
                for(var k = 0; k < blockSize; ++k) {
                    value += first.getAt(i, firstCol + k) * second.getAt(secondRow + k, j);
                }
                result.setAt(result.getAt(i, j) + value, i, j);
            }
        }
    }
}


// ./Lab8/Server/src/main/java/org/example/Algorithms/FoxAlgorithm.java

package org.example.Algorithms;

import org.example.LabMath.Matrixes.Matrix2D;
import org.example.LabMath.Matrixes.Matrix2DFactory;

public class FoxAlgorithm extends GeneralAlgorithm {
    private static final String ERROR_PROCS_NUM = "Number of processes must be
a square number";
     private static final String ERROR_SQUARE_MATRIX = "Rows and columns are
not equal";
     private static final String ERROR_BLOCK_SIZE = "Matrix size must be divisible
by number of threads";
```

```
public FoxAlgorithm() {}

public FoxAlgorithm(int threadsNum, Matrix2D first, Matrix2D second) {
   super(threadsNum, first, second);
}

public static void main(String[] args) {
   var matrixFactory = new Matrix2DFactory();
   var rows = 3;
   var cols = 3;
   var minVal = 0;
   var maxVal = 10;
   var threadsNum = 3;
   var seed = 0;

   var first = matrixFactory.getRandom(rows, cols, minVal, maxVal, seed);
   var second = matrixFactory.getRandom(rows, cols, minVal, maxVal, seed);

   var algorithm = new FoxAlgorithm(threadsNum, first, second);
   var striped = new BlockStripedAlgorithm(threadsNum, first, second);
   var result = algorithm.solve();
   var stripedResult = striped.solve();
   System.out.println("First:\t" + first);
   System.out.println("Second:\t" + second);
   System.out.println("Fox:\t" + result);
   System.out.println("Striped:\t" + stripedResult);
}

private void checkIfSquare(Matrix2D matrix) {
   if(!matrix.isSquare()) {
      throw new IllegalArgumentException(ERROR_SQUARE_MATRIX);
   }
```

```
}

@Override
public void setFirst(Matrix2D first) {
   checkIfSquare(first);
   super.setFirst(first);
}

@Override
public void setSecond(Matrix2D second) {
   checkIfSquare(second);
   super.setSecond(second);
}

@Override
public Matrix2D solve() {
   var rows = first.getRows();

   var blockSize = rows / threads.length;

   if(rows % threads.length != 0) {
      throw new IllegalArgumentException(ERROR_BLOCK_SIZE);
   }

   var matrices = new Matrix2D[threads.length];
   for(var i = 0; i < threads.length; ++i) {
      matrices[i] = new Matrix2D(rows, rows);
      threads[i] = new FoxAlgorithmThread(i, blockSize, first, second, matrices[i]);
      threads[i].start();
   }

   for(var t : threads) {
```

```
            try {
                t.join();
            } catch(InterruptedException e) {
                throw new RuntimeException(e);
            }
        }

        if(threads.length == 1) {
            return matrices[0];
        }

        for(var i = 1; i < matrices.length; ++i) {
            matrices[0].add(matrices[i]);
        }

        return matrices[0];
    }
}


// ./Lab8/Server/src/main/java/org/example/Algorithms/GeneralAlgorithmThread.java

package org.example.Algorithms;

import org.example.LabMath.Matrixes.Matrix2D;

public abstract class GeneralAlgorithmThread extends Thread {
    protected Matrix2D first;
    protected Matrix2D second;
    protected Matrix2D result;

        public  GeneralAlgorithmThread(Matrix2D  first,  Matrix2D  second,  Matrix2D
result) {
        this.first = first;
        this.second = second;
        this.result = result;
    }

    @Override
    public abstract void run();
}


// ./Lab8/Server/src/main/java/org/example/Algorithms/BlockStripedThread.java

package org.example.Algorithms;

import org.example.LabMath.Matrixes.Matrix2D;

public class BlockStripedThread extends GeneralAlgorithmThread {
    private final int step;
    private final int firstRow;

    public BlockStripedThread(int firstRow, int step, Matrix2D first, Matrix2D
second, Matrix2D result) {
        super(first, second, result);
        this.firstRow = firstRow;
        this.step = step;
    }

    @Override
    public void run() {
        var firstRows = first.getRows();
        var firstCols = first.getCols();
        var secondCols = second.getCols();
        var curRow = firstRow;
        while(curRow < firstRows) {
            for(var j = 0; j < secondCols; ++j) {
                var value = 0.0;
                for(var k = 0; k < firstCols; ++k) {
                    value += first.getAt(curRow, k) * second.getAt(k, j);
                }
                result.setAt(value, curRow, j);
            }
            curRow += step;
        }
    }
}


// ./Lab8/Server/src/main/java/org/example/Algorithms/GeneralAlgorithm.java

package org.example.Algorithms;

import org.example.LabMath.Matrixes.Matrix2D;

public abstract class GeneralAlgorithm {
     protected static final String ERROR_MULTIPLICATION = "Rows and columns
are not equal";
    protected static final String ERROR_NUM_OF_THREADS = "Number of threads
must be positive";

    protected Thread[] threads;
    protected Matrix2D first;
    protected Matrix2D second;
    public GeneralAlgorithm() {}

    GeneralAlgorithm(int threadsNum, Matrix2D first, Matrix2D second) {
        setThreadsNum(threadsNum);
        setFirst(first);
        setSecond(second);
    }

    public void setThreadsNum(int threadsNum) {
        if(threadsNum <= 0) {
            throw new IllegalArgumentException(ERROR_NUM_OF_THREADS);
        }
        if(this.threads != null && this.threads.length == threadsNum) return;
        this.threads = new Thread[threadsNum];
    }

    public void setFirst(Matrix2D first) {
        this.first = first;
    }

    public void setSecond(Matrix2D second) {
        this.second = second;
    }

    public abstract Matrix2D solve();
}


// ./Lab8/Server/src/test/java/test/LabApplicationTests.java

package test;
```

```
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
public class LabApplicationTests {
        @Test
        void contextLoads() {}
}



// ./Lab8/Server/src/test/java/test/MatrixTester.java

package test;

import com.github.sh0nk.matplotlib4j.*;
import org.example.LabMath.Matrixes.Matrix2DFactory;
import org.example.Algorithms.GeneralAlgorithm;

import java.io.*;
import java.util.ArrayList;

public class MatrixTester {
   private static final int minVal = -10;
   private static final int maxVal = 10;
   private static final String FILE_NAME = "results.csv";
   private static final String DELIMETER = "\t";
   private static final String LEGEND_POSITION = "upper left";
   private static final String X_LABEL = "Matrix size";
   private static final String Y_LABEL = "Milliseconds";
   private static final String SPEEDUP_Y_LABEL = "Speedup";
   private static final String PLOT_FILE = "plot.png";
   private static final String SPEEDUP_PLOT_FILE = "speedup_plot.png";
```

```
    private static class AlgorithmResult {
        public final long milliseconds;
        public final int threadsNum;
        public final int size;
        public final String name;
        public final double speedup;

            public AlgorithmResult(long time, int threadsNum, int size, double speedup,
String name) {
            this.milliseconds = time;
            this.threadsNum = threadsNum;
            this.size = size;
            this.name = name;
            this.speedup = speedup;
        }

        @Override
        public String toString() {
                return String.format("%s\t%d\t%d\t%d\t%f", name, threadsNum, size,
milliseconds, speedup);
        }
    }

        public static void main(String[] args) throws PythonExecutionException,
IOException {
            var tester = new MatrixTester();
//          var threadsNums = new int[] {1, 4, 10};
//          var matrixSizes = new int[] {100, 200, 300, 400, 500, 600, 700};
//              var algorithms = new GeneralAlgorithm[] {new FoxAlgorithm(), new
BlockStripedAlgorithm()};
```

```
//          tester.testAlgorithm(threadsNums, matrixSizes, algorithms);

        tester.plotStatistic();
        tester.plotSpeedup();
    }

    public void plotStatistic() throws PythonExecutionException, IOException {
        Plot plt = Plot.create();
        var results = readStatistic();
        var algorithms = results.stream().map(r -> r.name).distinct().toList();
        var threadNums = results.stream().map(r -> r.threadsNum).distinct().toList();
        for(var a : algorithms) {
            var filtered = results.stream().filter(r -> r.name.equals(a)).toList();
            for(var threadNum : threadNums) {
                    var filteredByThreadNum = filtered.stream().filter(r -> r.threadsNum ==
threadNum).toList();
                var x = filteredByThreadNum.stream().map(r -> r.size).toList();
                var y = filteredByThreadNum.stream().map(r -> r.milliseconds).toList();
                plt.plot().add(x, y).label(a + " " + threadNum);
            }
        }
        plt.legend().loc(LEGEND_POSITION);
        plt.xlabel(X_LABEL);
        plt.ylabel(Y_LABEL);
        plt.savefig(PLOT_FILE);
        plt.show();
    }

    private void plotSpeedup() throws IOException, PythonExecutionException {
        Plot plt = Plot.create();
        var results = readStatistic();
        var algorithms = results.stream().map(r -> r.name).distinct().toList();
        var threadNums = results.stream().map(r -> r.threadsNum).distinct().toList();
        for(var a : algorithms) {
            var filtered = results.stream().filter(r -> r.name.equals(a)).toList();
            for(var threadNum : threadNums) {
                    var filteredByThreadNum = filtered.stream().filter(r -> r.threadsNum ==
threadNum).toList();
                var x = filteredByThreadNum.stream().map(r -> r.size).toList();
                var y = filteredByThreadNum.stream().map(r -> r.speedup).toList();
                plt.plot().add(x, y).label(a + " " + threadNum);
            }
        }
        plt.legend().loc(LEGEND_POSITION);
        plt.xlabel(X_LABEL);
        plt.ylabel(SPEEDUP_Y_LABEL);
        plt.savefig(SPEEDUP_PLOT_FILE);
        plt.show();
    }



    private ArrayList<AlgorithmResult> readStatistic() throws IOException {
        var line = "";
        BufferedReader br = new BufferedReader(new FileReader(FILE_NAME));
        var results = new ArrayList<AlgorithmResult>();
        while ((line = br.readLine()) != null) {
            String[] row = line.split(DELIMETER);
            results.add(
                new AlgorithmResult(
                    Long.parseLong(row[3]),
                    Integer.parseInt(row[1]),
                    Integer.parseInt(row[2]),
                    Double.parseDouble(row[4]),
                    row[0]
                )
            );
        }
        return results;
    }

    public void testAlgorithm(int[] threadNums, int[] matrixSizes, GeneralAlgorithm[]
algorithms) throws IOException {
        var file = new File( FILE_NAME);
        var results = new FileOutputStream(file);
        var matrixFactory = new Matrix2DFactory();
        for(var algorithm : algorithms) {
            var algName = algorithm.getClass().getSimpleName();
            for(var size : matrixSizes) {
                long threadTimeOne = 0;
                for(var threadsNum : threadNums) {

                    var startTime = System.currentTimeMillis();

                    var first = matrixFactory.getRandom(size, size, minVal, maxVal, 0);
                    var second = matrixFactory.getRandom(size, size, minVal, maxVal, 0);

                    algorithm.setThreadsNum(threadsNum);
                    algorithm.setFirst(first);
                    algorithm.setSecond(second);

                    algorithm.solve();

                    var endTime = System.currentTimeMillis();

                    var duration = endTime - startTime;
                    if(threadsNum==1) threadTimeOne = duration;

                    var result = new AlgorithmResult(
                            duration, threadsNum, size,
                            threadTimeOne / (double) duration , algName).toString();

                    System.out.println(result);
                    results.write((result + "\n").getBytes());
                }
            }
        }
    }
}
```

```
// ./Lab8/Client/main.py


import requests
import json
import random
import time
from enum import Enum, auto

URL_CLIENT_MULTIPLY = 'http://localhost:8080/api/clientMultiply'
URL_SERVER_MULTIPLY = 'http://localhost:8080/api/serverMultiply'
HEADERS = {"Content-Type": "application/json; charset=utf-8"}

class AlgType(Enum):
  BLOCK_STRIPED = auto()
  FOX = auto()
  NATIVE = auto()
```

```
def gen_list(size: int):
    return [random.uniform(0, 1) for _ in range(size * size)]

def client_multiply(algType: AlgType, size: int, threadsNum: int):
    data = {
        'algType': algType.name,
        'first': {
            'rows': size,
            'cols': size,
            'values': gen_list(size)
            },
        'second': {
            'rows': size,
            'cols': size,
            'values': gen_list(size)
            },
        'threadsNum': threadsNum
        }
    return requests.get(URL_CLIENT_MULTIPLY, headers=HEADERS, json=data)

def server_multiply(algType: AlgType, size: int, threadsNum: int):
    data = {
        'algType': algType.name,
        'rows': size,
        'cols': size,
        'threadsNum': threadsNum
        }
    return requests.get(URL_SERVER_MULTIPLY, headers=HEADERS, json=data)
```

```
sizes = [s for s in range(30, 390, 30)]
threads = [s for s in range(2, 5)]
funcs = ['client_multiply', 'server_multiply']

lines = []
for f in funcs:
    for s in sizes:
        for t in threads:
            nanoseconds = time.time_ns()
            if f == funcs[0]:
                client_multiply(AlgType.BLOCK_STRIPED, s, t)
            else:
                server_multiply(AlgType.BLOCK_STRIPED, s, t)
            nanoseconds = time.time_ns() - nanoseconds
            millisecs = float(nanoseconds) / 10e6
            res_str = f'{f}\t{s}\t{t}\t{millisecs}\n'
            print(res_str, end='')
            lines.append(res_str)

with open('log.log', 'w') as file:
    file.writelines(lines)
```

```
// ./Lab8/Client/build_graph.py

import matplotlib.pyplot as plt
import pandas as pd

vals = []
with open('log.log', 'r') as file:
    for line in file.readlines():
        vals.append(line.split('\t'))

df = pd.DataFrame(vals)
df.columns = ['rtype', 'matsize', 'tnum', 'millis']
for col in ['matsize', 'tnum']:
    df[col] = df[col].astype('int')

for col in ['millis']:
    df[col] = df[col].astype('double')

rtype = list(set(df.rtype.to_list()))
tnum = list(set(df.tnum.to_list()))

fig, ax = plt.subplots(1, 1, figsize=(7, 7))
for tb in rtype:
    for tn in tnum:
        dd = df[(df.rtype == tb) & (df.tnum == tn)]
        plt.plot(dd.matsize, dd.millis, label=f'{tb} {tn}')

plt.ylabel('Milliseconds')
plt.xlabel('Matrix size')
plt.legend(loc='upper left')
plt.savefig('plot.png')
#plt.show()
```